International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 3, Issue 5, Dec 2013, 1-10

AN EFFICIENT DEBUGGING TOOL FOR OBJECT ORIENTED SYSTEM

D. M. THAKORE[1]& TANVEER S BEG[2]

[1]Guide, Bharati Vidyapeeth Deemed University College of Engineering, Pune, Maharashtra, India

[2]Research Scholar, Bharati Vidyapeeth Deemed University College of Engineering, Pune, Maharashtra, India

ABSTRACT

Inappropriate debugging techniques during software development may leads to some blunder mistakes in later stages of software development, due to which intended and projected functionality of the software is difficult to achieve. In the process of Software Development and evolution, Developer has to answer multiple questions about how the code or software behaves at runtime. The traditional or classical debugger while debugging gives developer bunch of breakpoints in the source code. This is an imprecise and inconsistent stage which is difficult to be used by the developer for development. Some of debugging tools are helpful for understanding the problems, as stated from traditional tools that the complexity of object oriented system expands, debugging becomes considerably difficult. Developer needs a dedicated user interface for these operations on objects; this need is fulfilled by facilitating a user interface for the programmer.

Object based debugging tool looks forward to analyze the relationship in between the objects during the runtime. There exists therefore conceptual gap between the interface offered by the debugger and the need of the developer, hence to overcome this drawback or problem; there is a need for object based debugger and useful interface for it. In this paper, reviewing different existing tools for analyzing debugging tool requirements in context of programmer point of view, So that programmer can get clear idea about intended functionality of developing software.

KEYWORDS: Software Programming, Debugging, Objects, UI, Errors

INTRODUCTION

Traditional or Classical Debugger always used to concentrate on the execution of stack. In this scenario programmer recognises parts of source code of interest and sets breakpoints accordingly. These breakpoints are set of purely with respect to static abstraction and not respect to particular object of the running system. This Object based debugging as an alternative approach to interacting with a running software system. Mainly Focusing on objects as the key point, natural debugging operations can be defined to answer developer questions related to runtime behaviour. Here, presented scenario of an object based debugger. How it offers more effective support for many typical developer tasks than a traditional or classical debugger.

Traditional or Classical debuggers are not always up to the task, since they bind to provide access to information that is still in the run-time stack. In some cases, the information needed to track down these difficult bugs content; how an object reference got here, the previous values of object's fields. For this reason it is helpful to have previous object states and object reference flow information at hand during debugging.

RELATED WORK

Debugging Tools

Now day's lots of debugger tools are present. The capabilities of some of the available debuggers some of them explained and discussed. Such as jBixbe, DBG | PHP, Jswat, Backstop tool, WPOL, CMeRun, CnC and OOCD. Bixbe apply

this in Java applications on the theoretical level at which they are designed it possible to find simple bugs as well as weaknesses or insufficiencies in application given. It shows the details of application selecting particular classes, objects, with their relationships. jBixbe provides a new feature of debugging complex Java applications by allowing their structure and functioning on the conceptual stage of the UML. This debugger also explains object-oriented concepts and provides source code debugging and breakpoints. But there are few limitations of these debuggers. Some researches felt that it is complex when debug a large application because jBixbe is made for high level object-oriented Java debugger. That's why it is very difficult for entry level learner to study or understand the error because the jBixbe can't locate the error. This debugger do not have pop-up window to tell a user what to do if errors occurs. The next example is a DBG | PHP–Debugger DBG (NuSphere, 2009) which is an open source debugger profiler for PHP programming language. PHP Debugger which is best tool for helping the bugs fast and eliminates them from the PHP programs. It supports not only a GUI interface but also command-line interface. DBG acts as full-featured PHP debugger, an interactive tool, helps debugging PHP scripts.

It acts on development web server, allows debug your scripts locally or remotely, from an IDE or console. PHP Debugger provides a powerful as well as easy way to simplify PHP debugging because it allows complete visibility as well as control over the execution of PHP scripts. This debugger also doesn't require that you make any changes to your PHP code. PHP Debugger can be debugging PHP applications on eighteen different platforms locally and remotely. Benifits for the debugging of nested calls multiple debug sessions as well as debugging of PHP CLI scripts set PhpED apart from other PHP IDE's. The advantages of DBG| PHP Debugger are allow user step by step through the execution of a PHP scripts, line-by-line as well as user friendly GUI. It also has good representation of data structures and have the call stack window displays the function call that brought user to the current script location as well as allows multiple debugger processes running equally. Even though with benefit of that DBG | PHP-Debugger can give, but there are some researches implement need Object-oriented Java Debugger which cannot support by this debugger.

Another drawback is PHP is an old script or many of organization have change to JSP in web development. The next example is a Jswat debugger (Swat, 2009). Which is a standalone and a graphical Java debugger, made to use the Java Platform Debugger Architecture. The Jswat features, breakpoints with conditionals as well as monitors actions, source code display, graphical display showing all threads, stack frames, visible variables as well as loaded classes, command interface for advanced features or Java-like expression evaluation including method invocation.

The benifits of Jswat debugger which are simple and user friendly GUI. This debugger is best for analyzing applications (maintenance) or display object relationships (structure diagrams). Even with features that Jswat debugger can give, it also have some drawbacks such as do not have pop-up window to tell a user what to do in step by step through the execution of a PHP scripts or do not show as a line by line but it just only pin-pointed on the specific error. Backstop tool (Murphy *et al*., 2004), select the common runtime error in Java applications or do not identify the logic error. This tool devoloped for programmers studying Java at the entry level as well as it provides more user-friendly error messages when an uncaught runtime error (exception) occurs. It also gives the debugging support by users to watch the execution of the program and the changes to the values of variables. Same way with the educational tool Expresso proposed by Hristova *et al*. (2003) select or identifies the common error in Java programming and generates error messages which provide suggestions on how to fix the code. Those existing tools have been designed to identify logic errors faster but not give any suggestions to resolve them.

There also has been investigation of debugging techniques among the entry level programmers. The tool that presented in this research can be used to do debugging techniques among novice stable or not fragile. CAP (Schorsch, 1995), developed to aid entry level programmers in a user-friendly fashion by getting all syntax logic as well as style errors that they

make in Pascal program. Which also supply information to the beginners about the error, the reason error has been occur as well as give the solution to fix the problem. According to Ebrahimi and Schweikert (2006), novices may not detect negative interactions between section and block of code. For example, the code to perform the output which is correct but in the wrong place in the program. So WPOL (Ebrahimi and Schweikert, 2006) designed to facing the problem. WPOL being designed to incorporate the Plan-Object-Paradigm, Web as well as assessment with focus on plan integration. WPOL is plan object-oriented teaches novices programming by plan management as to how they are integrated as well as bridges the gap between object and functions. A plan that used in WPOL is an abstraction of a concept, requirement, object and programming code. The plan is used for structured knowledge representation in natural language processing. Another debugger is Check 'n' Crash (CnC) introduced by Csallner and Smaragdakis (2005) is an automatic error detection which combines the static checking as well as automatic test generation to get the best of both function in order to detecting errors. The CnC tool combines the advantages of ESC/Java static checker and JCrasher tool automating testing tool. This tool, it consist of taking the abstract error conditions using theorem proving techniques by a ESC/Java static checker and deriving the specific error conditions using a constraint solver which then produce concrete test cases that are executed to determine whether an error truly exists by JCrasher tool. Visual tool which is an alternative ways, which help the novices more understandable when learning the programming language.

It can show for beginners what happens when the code is executed. Visual debugger for Java programs (JVD) introduced by Rafieymehr and McKeever (2007) is developed using the graphical animation as well as runtime state retention to display program state during execution. These functions to detect runtime errors by determine which classes have main methods as well as ask user to choose one and display the code with highlight showing current line. The code which is displayed in typical balloon type boxes. Giving compiler type error messages is challenging for novice programmers (Hartmam *et al*., 2010). HelpMeOut by Hartmam *et al*. (2010) is a tool that aids novices with the debugging of compiler error messages by giving successful solution to the errors that other programmers have got. Many tools have been designed in debugging area to find bugs in software but some of available technique are difficult to use and not benefits in finding real bugs. In scenario to study in depth of programming processes there have two types of vital thing. In the beginning they control the knowledge structures that programmers possess if they wish to measure the effects of factors that influence programmer performance. Another is, the researchers should understand the knowledge structures that novice programmers possess (Vessey, 1985).

This research has concentrated on the problems that occur in debugging process to object-oriented programming among the novice programmers. Next discuss the ways that can be used to improve the learning of object oriented programming. Data Debugging in this given is an approach for locating potential data errors. When it is impossible to know whether data are full of errors or not, data debugging does the next best thing: locating data that has an unusual impact on the computation. Intuitively, data that has vital impact on the last result very important, By contrast, wrong data whose presence has no particularly unusual effect on the final result does not merit special attention. Data debugging get together data dependence analysis and statistical analysis to find as well as rank data based on the unusualness of its impact on the results of a computation.

- Data entry errors, including typographical errors and transcription errors from illegible text.
- Measurement errors, when the data source itself, such as a disk or a sensor, is faulty or corrupted (unintentionally or not).

- Data integration errors, where inconsistencies arise due to the mixing of different data, including unit of measurement mismatches. While data errors pose a threat to the correctness of any computation, they are especially problematic in data-intensive programming environments like databases, spreadsheets, and certain scientific computations. Back-in-time debuggers approach. These are extremely useful tools for identifying the causes of bugs. Compare to the "omniscient" approaches that try to remember all previous states are impractical because they consume too much space as well as they are too slow. So many techniques to limit these demerits, but they eventually end up giving out too much relevant data.

In this paper a practical approach that attempts to keep track of only the relevant data. In contrast to other approaches, it keeps object history information together with the regular objects. This method has the effect that data not reachable from current application objects that's why not useful further. This approach, present idea which explains that memory utilization stays in practical limits. Furthermore, the performance penalty is significantly less than with other approaches [1].

Back-in-Time Debugging: Back-in-Time Debuggers are useful tool for identifying the cause of errors not the omniscient debugger which always remembers all previous states [24]. Some features and demerits are shown in table 1 as shown.

To overcome this drawback of omniscient debugger back in time debugger is developed. Omniscient Debugging: also known as back-in-time debugging or reversible debugging. These debuggers store the total history and execution trace of a debugged program. Developers can explore the history by simulating step-by-step execution both forward and backward [1] [6]. Query Based debugging approach**.** User defines a query in a higher-level language that is then applied to the data Queries can test complex object interrelationships and sequences of related events. Trace oriented Debugger: it is collected of a well-organized instrumentation for incident making, a specific database for scalable storage space as well as support for partial traces which reduce trace volume [2]. While this method has the advantage that nowhere data is lost, its drawback is that it requires large hardware power, which is not available for many developers today [6]. The why line debugging interface approach. Why line tool which benefits developer to ask about, "Why did" as well as "Why did not" questions with their program's output Why line tries to facilitate developer by applying static as well as dynamic analyses and after that answer Some of the developer questions [7]. Auto Flow an automatic debugging approach. Aspect-oriented programming (AOP) is gaining popularity with languages such AspectJ. when AspectJ software evolution, tests fail, it may be lengthy or difficult for programmers to find out the failure minimizing changes by manually inspecting all code editing.

To beat the costly attempt spent on debugging developed AutoFlow, an automatic debugging approach for AspectJ system. AutoFlow meets the potential of delta debugging algorithm with the benefit of change impact analysis to slow down the search for imperfect changes. It primary uses change collision analysis to identify a subset of responsible changes for a failed test, after this ranks these changes according to proposed heuristic (indicating the likelihood that they may have contributed to the failure), finally this improved delta debugging algorithm to determine a minimal set of faulty changes. The important advantage of AutoFlow is that it can automatically reduce a big portion of irrelevant change in an early stage, eventually then locate not fixed changes effectively [8]. How helpful are automated debugging tools: The Area of automated debugging, which is with the automation of identifying and correcting a failure's root cause, made tremendous advancements in the past years. These assumptions concern the work process of developers and their ability to detect wrong code without explanatory context, or the size and arrangement of fixes. Instead of trying to locate the fault, this proposes to help the developer understand it, thus enabling her to decide which fix they deems most appropriate.

This came to know the need to employ a completely different evaluation scheme that bases on feedback from actual users of the tools in realistic usage scenarios [9]. NUDA a Non-Uniform Debugging approach this paper is proposed a novel non-uniform debugging architecture (NUDA). This makes hardware-assisted debugging both feasible and scalable for many-core processing scenarios. Here, theme is to distribute the debugging support structures across a set of hierarchical clusters while avoiding address overlap. It allows the address space to be monitored using non-uniform protocols and propose approach to lockset-based race detection supported by the NUDA. Here, page-based monitoring cache in every NUDA node to keep track of footprints [10]. "A Review of reverse debugging" defined as of a debugger to stop after a failure in a program has been observed and go back into the history of the execution to find reason for the failure. New approach for object based debugger is shown in figure 1. Reverse execution has become a practical technique available in a number of free and commercial tools. This article review the history and techniques of reverse debugging, as researched, implemented, and used until today [11].

There is a need to find or steer in area where programmers actually face problems during debugging scenario [12][24][26]. This strategy works well, trying to understand the general performance for objects. In these scenarios need an object-specified analysis and simple breakpoint strategy is not the best option. In application development when programmers require interrupting the execution of the application when a particular code is evaluated, requires breakpoint strategy. The programmer wants to locate the particular object he is concerned. The programmer explains a proper condition to recognize the particular object previously found, without interacting with it. This approach may be realistic, if exist few objects to determine in given code [13] [24].

Table 1: Table Various Debugging Techniques

Factors / Capabilities	jBixbe	DBG PHP	Jswat	Backstop Tool	WPOL	CAp	CnC
Year	2006	2009	2009	2004	2006	1995	2005
Concentrated on	java	Php programs	Graphical java debugger	java	Plan object paradigm	pascal	Java
Working plattform	java	php	java	java	java	pascal	Java
Linebyline Execution	✓	✓	✗	✓	✓	✓	✓
Execution visible	✗	✓	✓	✓	✗	✗	✗
Identifies Logic error				✗		✓	✓
Web navigation available	✗	✗	✗	✗	✗	✗	✗
Common syntax logic and style errors	✓	✓	✓	✓	✓	✓	✓
Class object interaction/relationship	✓		✓	✗	✗	✗	✗
User interface friendly	✓	✓	✓	✗	✗	✓	✗
breakpoints	✓		✓	✓	✓	✓	✓
Answer questions regarding with	Application design, data structures	Php applications, data structures	Application maintanance	Except logic error	Structured knowledge representation	Common syntax, logic and style	Abstract error conditions
Popup window	✗		✗	✗	✗	✗	✓
Complicated in case	Large applications	No support new tech	Complex code	Logic errors	Complex code	Other platforms	Derive specific error
Remark	Do not locate error	Php is old script	Require popup window	do not identify the logic error	incorporate the Plan-Object-Paradigm	Syntax logic style Errors in pascal	Produces test cases for finding error

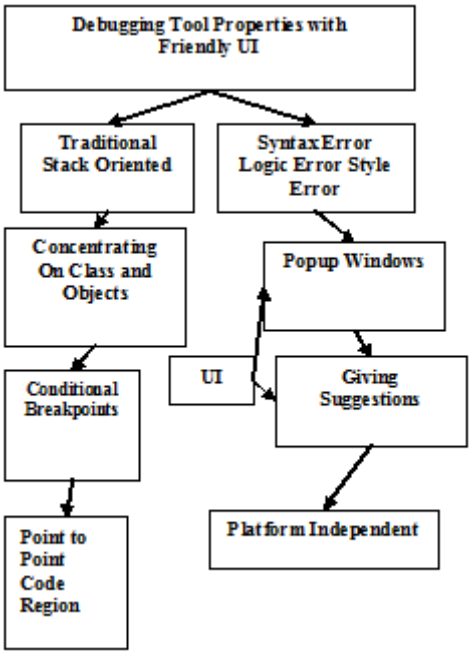

Figure 1: Object Based Debugging Tool

ISSUES AND CHALLENGES

Studding and analyzing different literature survey following are the outcomes.

- Back in time debugging debugger have to remember history of all previous states.
- Trace oriented debugger requires more hardware power, which is practically not possible. Omniscient debugger depend on more memory because, to store history of last stages. Reverse debugging is to stop after a failure in a program has been observed and go back into the history of the execution to uncover the reason for the failure.
- AutoFlow used to reduce a large portion of irrelevant change in an early phase, finally then locate faulty changes effectively.
- After going through literature survey came to know that developer faced some kind of problems while doing debugging.
- Major problem is that developer cannot answers about objects. After analyzing on problems faced by developer they do not find out answer to their question in case of object.
- There is also a good scope of a useful and dedicated user interface for debugging scenario.
- Developer will be comfortable with using object oriented dedicated user interface for debug situations.

When complex object oriented system taken in account then traditional debuggers fails to act on object related operations and relationship between different objects.

To overcome these problems new tool should be developed on object based approach and useful dedicated user interface for it.

MOTIVATION SCENARIO

The motivation for doing this project was primarily an interest in undertaking a challenging project in an interesting

area of debugging which gives opportunity to learn about new area of software engineering. This area is possibly an area that I might study at postgraduate level. As the debugging area taken into account developer came across different problems, which are faced by them. The traditional debugging technique used by programmer is concentrated on stack orientation so developer face problems regarding objects in the code given. The debuggers not designed to answer many of the questions that developer typically uses to ask after analyzing different papers related to approaches of debugging, found that one can develop a debugging tool which is based on objects, and possesses following some points to understand runtime behavior of the system. It will be helpful to continue interacting with the runtime, applying operations directly to objects without working with static representation of the system. This is useful in to monitor communications with entity objects without taking timely advanced breakpoints. Now which required specifies scenario to develop object based debugging tool that facilitated with user interface which fulfill needs of developer such as, different interruption related to objects or keep watch on object interactions and do operations related to objects using user interface telling suggestions.

CASE STUDY

We have considered here object based debugging tool for case study. Our object based debugging tool system has its unique environment diagram as follows;

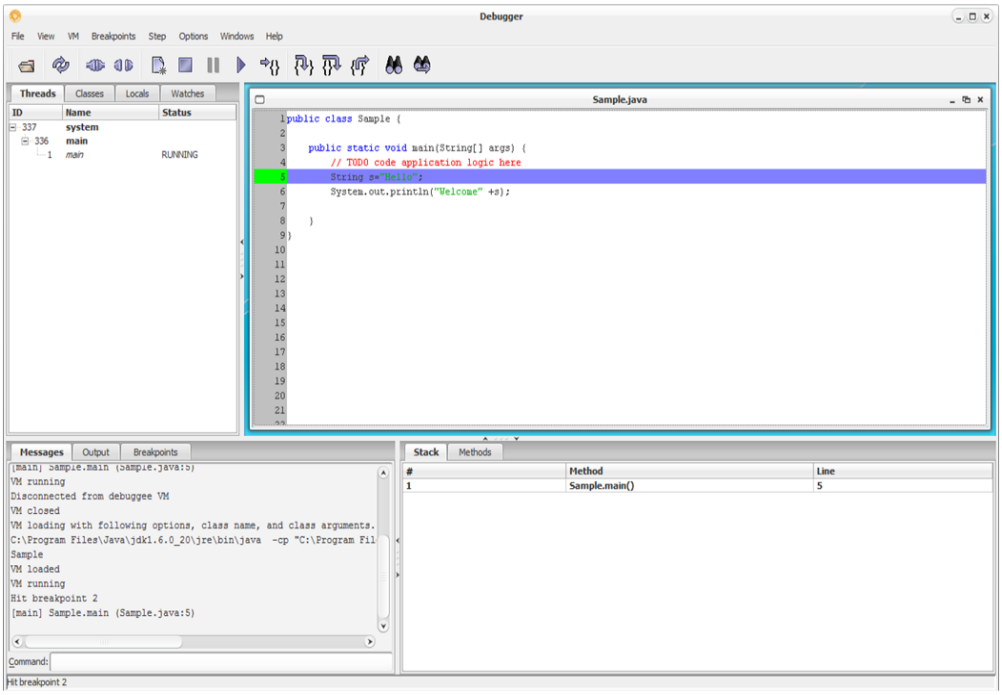

Figure 2: Screenshot Shows the Environment Details for the Given Input Example

CONCLUSIONS

While the ultimate goal may be to develop code that has zero issues, software engineers must live in the real and imperfect world where system complexity is important thing will going to occur. Due to this The fact that debugging remains the most problematic and economical costly issue of the development cycle indicates that engineers must digest the idea of day to day increasing complexity, prepare for the challenge from day one and acquire more as well as better debugging tools to keep increasing demands on developments.

Programmers have been waiting a long time for practical automated debugging tools, they already gone a long way from near the beginning days of debugging to further advance the state of the art in this scenario, must push research towards more capable directions that take into account the way programmers actually debug in real scenarios.

ACKNOWLEDGMENTS

I am very grateful to the people those who have provided me continuous encouragement and support to all the stages and ideas visualize. I am very much grateful to the complete BVDU group for open handed me all facilities and work environment which enable me to complete my task. I express my sincere thanks to Prof.Dr.D.M.Thakore, Prof.Dr.S.H.Patil, Head of the Computer Department, BVDU College of Engineering Pune who gave me their valuable and rich guidance and help in presentation of this research paper.

REFERENCES

1. Adrian lienhard, tudor Girba and OscarNierstrasz, ”Practical Object Oriented Back-In-Time Debugging”LNCS 5142, pp 592-615.
2. Raimondas Lencevicius, Urs Holzle and Ambuj K. Singh, “Query-based Debugging of Object-Oriented Programs” OOPSLA 97 Atlanta, USA.
3. Mark Minas “Cyclic Debugging For pSather, a Parallel Object-Oriented Programming Language” Jan 31 2002.
4. Tanja Mayerhofer, ”Testing and Debugging UML Models Based On Fuml” ICSE 2012.
5. G. Pothier, E. Tanter, and J. Piquer, “Scalable omniscient Debugging, “Proceedings of the 22nd Annual SCM SIGPLAN Conference on Object-Oriented Programming Systems, Languages And Applications (OOPSLA’07), vol. 42, no. 10, pp.535–552, 2007.
6. C. Hofer,M. Denker, and S. Ducasse, “Design and implementation of a backward-in-time debugger,” in Proceedings of NODE’06, ser. Lecture Notes in Informatics, vol. P-88 (GI), Sep 2006, pp. 17-32.
7. J. KO and B. A. Myers, “Designing the whyline: A debugging interface for asking questions about program behavior,” in Proceedings of the 2004 conference on Human factors in computing systems. ACM Press, 2004, pp. 151–158.
8. Sai Zhang; Zhongxian Gu; Yu Lin; Jianjun Zhao “AutoFlow: An automatic debugging tool for AspectJ software” ICSM 2008. IEEE International Conference on 2008, pp. 470 – 471.
9. Rossler, J. “How helpful are automated debugging tools?” User Evaluation for Software Engineering Researchers (USER), 2012 IEEE Conference Publications, pp. 13 – 16.
10. Chi-Neng Wen;shu-hsuan Chou;chih Chen ;tien-fu chen.”NUDA: A Non-Uniform Debugging Architecture and Nonintrusive Race Detection for Many core system” IEEE transaction, vol.61, 2012, pages.199-212.
11. Engblom, J “A Review of Reverse debugging” System, Software, SC and Silicon Debug Conference (S4D), 2012, pp. 1 – 6.
12. Chris parnin and alessandro orso, “Are automated debugging techniques actually helping programmers” ISSTA’ July 2011.
13. Jorge ressia, Alexandre Bergel and Oscar Nierstrasz “object centric debugging” ICSE 2012.
14. Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas and Carol Zander “Debugging: a review of the literature from an educational perspective” June 2008.
15. K. Glerum, K. Kinshumann, S. Greenberg, G. Aul, V. Orgovan, G. Nichols, D. Grant, G. Loihle, and G. Hunt,

"Debugging in the large: ten years of implementation and experience," Proc. SOSP, 2009, pp. 103-116.

16. Noor Fazlida Mohd Sani, Noor Afiza Mohd Arifin and Rodziah Atan "Design of object-oriented debugger model using unified modeling language" JCSSP 2013, pp 15-18.
17. Prof. D.M.Thakore, Torana N.Kamble "Validating UML Diagram for Security" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 2, Issue 5 pp.1574-1577.
18. D. M. Thakore Ravi P.Patki "Extraction of Class Model from Software Requirement Using Transitional SBVR format at Analysis Phase" International Journal of Advanced Research in Computer Science Volume 3, No. 7, ISSN No. 0976-5697 Nov-Dec 2012.
19. Potanin, A., Noble, J., Biddle, R.: Snapshot query-based debugging. In: Proceedings of the 2004 Australian Software Engineering Conference (ASWEC'04), Washington, DC, USA, IEEE Computer Society (2004) 251.
20. P. Iyenghar, C. Westerkamp, J. Wuebbelmann, E. Pulvermueller, A Model Based Approach for Debugging Embedded Systems in Real-time, in 10th ACM international conference on Embedded Software, EMSOFT 2010, ACM, New York, NY, USA, 69-78.
21. Ahmadzadeh, M., Elliman, D., & Higgins, CNovice programmers: An analysis of patterns of debugging among novice computer science students. (2005). Inroads, 37(3), 84 88.
22. Robertson, T., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, J., Beckwith, L., Impact of interruption style on end-user debugging. In E. Dykstra Erickson & M. Tscheligi (Eds.), Proceedings of the 2004 conference on human factors in computing systems (pp. 287 294) et al. (2004).
23. Prof.D.M.Thakore, Tanveer S Beg "An Automatic Debugging Tool Extension for Object Oriented Softwares" International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-3, Issue-2, May 2013.
24. Prof. D. M. Thakore, S.J.Sarde "Assessing the Software Complexity and Security metrics from UML Class diagram" International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 Vol. 2, Issue 4 pp.585-587.
25. Prof. S. D. Joshi, Prof.S.P.Abhang, Amruta Amune, Dhanraj Deshpande, Tanveer Beg. "Analysis of textual requirements & automatic generation of UML diagrams' International Journal of Advances in Management, Technology & Engineering Sciences (IJAMTES) International Conference On Current Trends And Challenges In Management, Engineering, Computer Applications Vol.1, Issue 6. March 2012.